Beeldverwerken Lab 3

Max Bos
10669027

Haischel Dabian
10336230

May 9, 2017

This document contains answers to the theory questions and notes to the exercises belonging to Lab 3.

2 Finding Lines using the Hough Transform

It is possible to find straight lines in digital images, by first using an edge detector to obtain image points that are on the desired curve in the image space, and subsequently use the Hough Transform to perform groupings of edge points into object candidates by performing an explicit voting proceure over a set of parameterized image objects.

The Hough Transform is implemented as the function named `hough`. The function takes a grey scale image, constructs an edge map by applying the Canny Edge detector, and then constructs a Hough transform for finding lines in the image. For the Canny Edge detection, the built-in Matlab function `edge` is used. For each non-zero point (x_i, y_i) in the edge image, we can ask what lines pass through that point. All such lines will satisfy an equation of the form:

$$x_i sin(\theta) - y_i cos(\theta) = \rho$$

The formula to characterize a line given in section 8.3 of the Lecture Notes is:

$$z_1 cos\phi + z_2 sin\phi = r$$

The formula given in the assignment is:

$$xsin(\theta_i) - ycos(\theta_i) = \rho$$

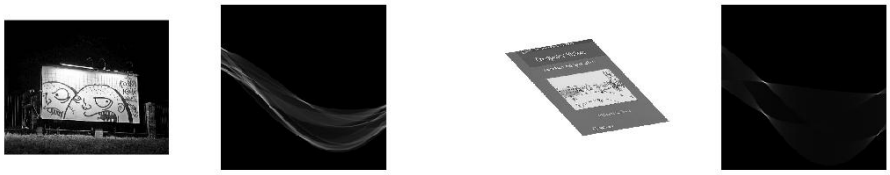
These formulas describe the same line mirrored over the Y-axis. Both formulas describe the same problem since both are capable of characterizing all lines in a 2 dimensional image.

ρ is the distance of the shortest line (perpendicular) from the origin to the line running through edge point (x_i, y_i). ρ depends on θ, where θ can be in the range of -90 and 90 degrees. ρ can be negative if θ is below zero.

The Hough Transform using `hough` has been performed on four different images. The results from these experiments can be seen in Figure 1.

(a) shapes.png

(b) box.png



(c) billboard.png

(d) szeliski.png

Figure 1: Hough Transform performed on the attachment images

3 Finding the Lines as Local Maxima

From the calculated Hough Transform of an image, it is possible to find the dominant lines and overlay them in the image. The function `houghlines` has been written, which finds the significant lines in an image using its Hough Transform. The displaying of the hough lines is done by the custom function `display_hough_lines`. `houghlines` returns the calculated significant lines. The drawing of the lines is not done inside the `houghlines` function.

The Hough Transform is first normalized before thresholding. Therefore, the threshold value should be between 0 and 1. The threshold would result in isolated regions that contain within them a peak indicating a maximal 'vote' for a line. Within each isolated region it finds the coordinates of the maximum value.

An option has been added to enable dilation on the Hough Transform before normalizing it. The structuring element of the dilation is a square shape of 2x2.

The implementation of `houghlines` has been tested on 'shapes.png'. To obtain the best possible set of lines, the threshold has been tweaked and image dilation is enabled. Figure 2 shows the result of the experiment.

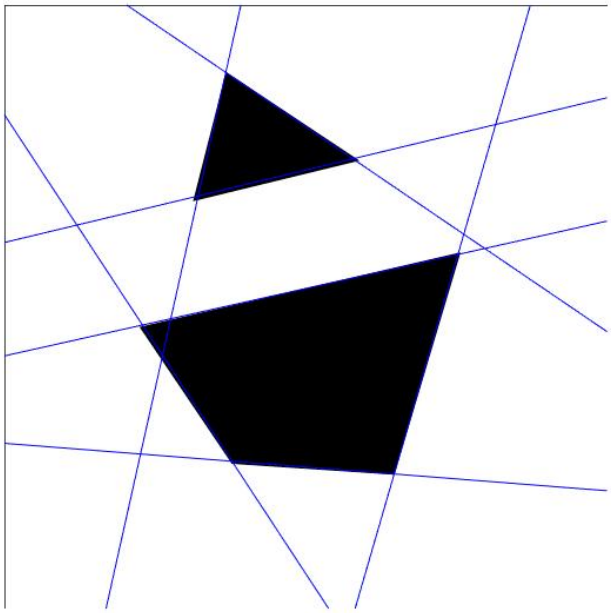


Figure 2: Hough lines of shapes.png

5 Optimal Line Estimation

The Hough method has returned an indication of where the dominant lines in the image are. But because of reasons of computational efficiency, we had to use rather coarse bins for the line parameters ρ and θ. The outcome of `houghlines` is used to select which points to group into a line, and from those the best line is determined using a least-square-fit. The selection of the points is done by `points_of_line` which collects the points close to a line found by the Hough transform, within a distance of epsilon. The estimation of the best line is done by `line_through_points`. The two functions have been grouped in one function named `optimal_lines_estimation`.

The implementation of `optimal_lines_estimation` is tested on 'shapes.png'. The same parameters are used for the test on houghlines. Figure 3 shows the result of the experiment.

6 Using the Lines

The straight lines that are the result of performing the optimal line estimation after Hough Transform, can be used to perform a straightening transformation on (unstraightened) objects in images. This can be done in two methods. The first is to calculate the intersections of the straight lines, which depict the corner points of the object. The second method is directly using the lines to obtain the transformation matrix.

The calculation of the line intersections is implemented in `intersections`. This function returns all intersections that lie inside the image range. The function has been tested using

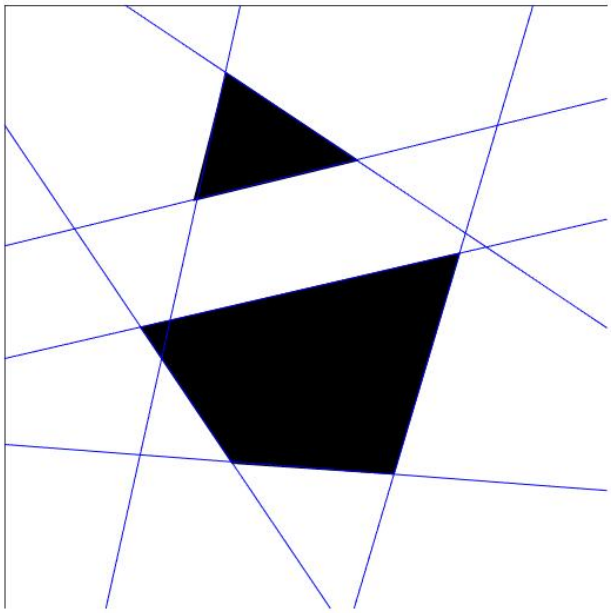
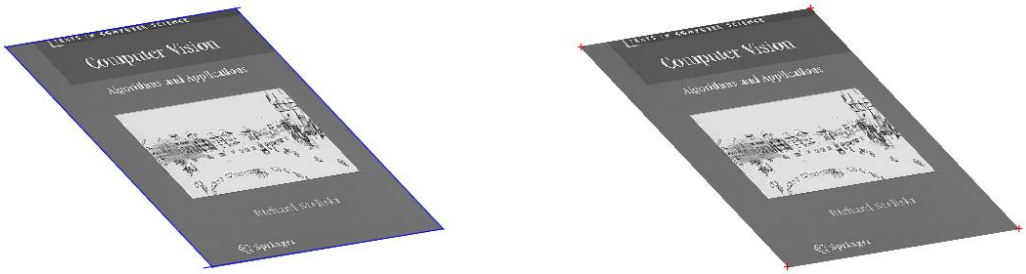


Figure 3: Optimal lines of shapes.png

'szeliski.png'. Figure 4 shows the result of the experiment.



(a) Optimal lines for szeliski.png

(b) Intersections of the optimal lines

Figure 4: Calculated intersections

Intersection of 2D lines is taking the cross product of their homogeneous representatives. This can be proved by the definitions of the cross product and the homogeneous weight. If we have the lines l_1 and l_2 their homogeneous weight is defined as:

$$\sqrt{l_{11} + l_{12} ... + l_{1n}}$$

$$\sqrt{l_{21} + l_{22} ... + l_{2n}}$$

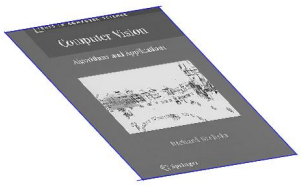
Their cross product is defined as:

$$l_1 l_2 = \|l_1\|\|l_2\| sin\theta$$

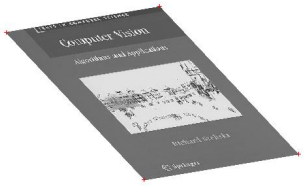
The representation of the homogeneous weight is equal to the magnitudes in the definition of the cross product. We multiply these magnitudes with $sin\theta$. So, the homogeneous weight that represents the intersection point is proportional to the sine of the angle.

It can be also shown that the cross product of two 2D points in their homogeneous representation is the homogeneous coordinate representation of the line connecting them. Following the same steps as before.

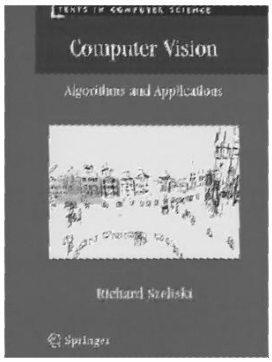
The Hough transform is used to detect lines and straighten some of the images in the attachment. The images that have been tested are 'szeliski.png' and 'billboard.png'. To obtain the optimal set of lines for the objects in the images, the thresholds have been tweaked. Subsequently, the intersections of the four lines for each image have been obtained, which depict the four corner points of the object in the image. The four corner points were matched to the resulting four corner points after transformation, and specified as input for the `myProjection` function. The result of the experiments can be seen in Figure 5 and Figure 6.



(a) Optimal lines for szeliski.png



(b) Intersections of the optimal lines



(c) Straightened flyer object

Figure 5: Straightening of szeliski.png

(a) Optimal lines for billboard.png (b) Intersections of the optimal lines (c) Straightened billboard

Figure 6: Straightening of billboard.png